High Spies
(or How to Win a Programming Contest)

André Deutz, Rudy van Vliet, and Hendrik Jan Hoogeboom

Leiden Institute of Advanced Computer Science (LIACS), Leiden University,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
`rvvliet@liacs.nl`

Abstract. We analyse transports between leaves in an edge-weighted tree. We prove under which conditions there exists a transport matching the weights of a given tree. We use this to compute minimum and maximum values for the transport between a given pair of leaves.

1 Introduction

You have been approached by a spy agency to determine the amount of contraband goods that are being traded among several nefarious countries. After being shipped from its country of origin, each container of goods is routed through at least one neutral port. At the port, the containers are stored in a warehouse before being sent on their way, so you cannot trace individual containers from their country of origin to their final destination. Satellite cameras can tell you the number of containers travelling in each direction on each leg of the journey. They cannot distinguish individual containers, nor do you have information on the times individual containers have been observed. You know, however, that every container takes the shortest possible route to its destination.

The task is to determine both the maximum and minimum number of containers that could have been travelling from one country to another. The transport network that is observed is an unrooted tree, with countries as leaves and ports as internal nodes. For each edge the number of containers is given in two directions. You know from the description above that no container leaves a port in the direction it came from.

This is the description of one of the problems at the Benelux Algorithm Programming Contest 2006, which was held in Leiden on 21 October 2006, see `www.bapc2006.nl`. The name of the problem was *High Spies*. The problem and some phrases in its description were taken from [Shasha, 2003].

We want to emphasize that High Spies is not just a maximum network flow problem, to which we can apply, e.g., the well-known Ford-Fulkerson algorithm (see [Ford and Fulkerson, 1957]). We do not so much consider networks with (maximum) capacities on the edges, but networks with numbers (of containers) on the edges that are *actually observed*, meaning that the numbers must really be met by the transport. Moreover, no container is allowed to travel from one node to another and back again.

P. Crescenzi, G. Prencipe, and G. Pucci (Eds.): FUN 2007, LNCS 4475, pp. 93–107, 2007.

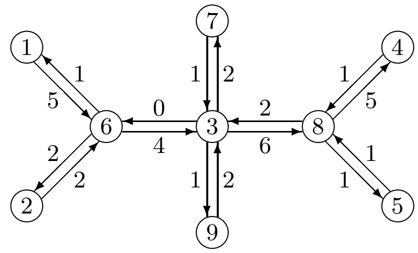

Fig. 1. A transport network with six countries and three ports

Consider, for example the network depicted in Fig. 1, and suppose we are interested in the transport from country 1 to country 4. The (standard) maximum flow from 1 to 4 consists of four containers, as this is the minimum number observed on the route from 1 to 4. It is, however, not possible to send four containers along this route. In that case, in port 6, one of the containers arriving from country 2 would be forced to return, which is forbidden.

Similarly, one might think that the minimum flow from 1 to 4 has value 0. Indeed, one may send the five containers leaving country 1 to countries 2, 5, 7 and 9. [1] However, in that case, at least one of the containers arriving at port 3 from port 8 would be forced to return.

In this paper, we present and analyse an algorithm for a generalized version of High Spies. In this version, the transport observed between two neighbouring nodes of the tree may be any non-negative real number.

The intuition behind the algorithm is simple. For each port on the (unique) route from a certain country (the source) to another country (the target), it determines the minimum and maximum number of containers that can be passed on from the direction of the source in the direction of the target. The minimum number results from passing on as many containers as possible in directions different from that of the target. In all this, we make sure that no container entering the port has to go back in the direction it came from. We use the (local) minima and maxima from the individual ports to compute the (global) minimum and maximum number of containers transported from source to target.

The paper is organized as follows. In Sect. 2, we model the problem in terms of weighted trees. In Sect. 3 and 4, we analyse which weighted trees actually correspond to valid transports. We will see that global conditions for this can be translated into easily checkable, local conditions. We need these conditions to justify our algorithm, which we describe in Sect. 5. There, we use the local conditions to determine the local maxima and minima just mentioned, and we combine these into a global solution. Finally, we make some concluding remarks, including some more remarks about the use of standard max-flow algorithms to solve the problem.

[1] Also this flow can be found with a standard max-flow algorithm, by introducing a special target node, which is only reachable from countries 2, 5, 7 and 9.

2 Problem Model

An unrooted tree can be denoted by an ordered pair $T = (V, E)$, where V is the (non-empty) set of nodes, and E is the set of edges of the tree. In High Spies, the direction of edges is important. Therefore, we consider an edge as an ordered pair of nodes (i, j). To reflect the undirected nature of the tree as a whole, we have $(i, j) \in E$, if and only if $(j, i) \in E$.

Definition 1. *A weighted tree is an ordered pair (T, c), where $T = (V, E)$ is an unrooted tree and c is a non-negative function (a weight function) on E.*

The weight $c(i, j)$ can be considered as the observed number of containers travelling from node i to node j. Note, however, that it does not have to be an integer number.

From now on, we assume that a tree $T = (V, E)$ contains at least two nodes. This allows for the identification of leaves and internal nodes in the tree. For ease of notation, we also assume that $V = \{1, \ldots, n\}$ for some $n \geq 2$. We let $\mathrm{Leaves}(T)$ denote the set of leaves of T.

Definition 2. *A transport Tr on an unrooted tree T is a non-negative function on the ordered pairs (l_1, l_2) with $l_1, l_2 \in Leaves(T)$ and $l_1 \neq l_2$.*

The number $\mathrm{Tr}(l_1, l_2)$ can be considered as the number of containers shipped from leaf (country) l_1 to leaf (country) l_2 via the edges of the tree. Despite this interpretation, $\mathrm{Tr}(l_1, l_2)$ does not have to be an integer number. Note that $\mathrm{Tr}(l_1, l_2)$ is not necessarily equal to $\mathrm{Tr}(l_2, l_1)$.

We are interested in the total transport over a certain edge (i, j) of the tree. This total transport comes from the leaves on one side of (i, j) and goes to the leaves on the other side of (i, j). We now define this formally.

Each edge (i, j) of the tree is a 'cut'. It partitions the set V of nodes into two subsets: the nodes on i's side of the tree, and the nodes on j's side of the tree. Let us call these subsets of nodes $\mathrm{Left}(i, j)$ and $\mathrm{Right}(i, j)$, respectively. This partitioning induces a partitioning of $\mathrm{Leaves}(T)$ into a subset of leaves on i's side of the tree, and a subset of leaves on j's side of the tree. Let us call these subsets $\mathrm{LLeaves}(i, j)$ and $\mathrm{RLeaves}(i, j)$, respectively. Clearly, $\mathrm{LLeaves}(i, j) = \mathrm{RLeaves}(j, i)$ and $\mathrm{RLeaves}(i, j) = \mathrm{LLeaves}(j, i)$.

For example, in the tree in Fig. 1, $\mathrm{LLeaves}(6, 3) = \{1, 2\}$ and $\mathrm{RLeaves}(6, 3) = \{4, 5, 7, 9\}$.

Definition 3. *A matching transport Tr on a weighted tree (T, c) with $T = (V, E)$ is a transport on T, such that for each edge $(i, j) \in E$,*

$$\sum_{l_1 \in LLeaves(i,j)} \; \sum_{l_2 \in RLeaves(i,j)} Tr(l_1, l_2) = c(i, j) \ . \tag{1}$$

Indeed, if we assume that the containers travelling from one leaf to another take the shortest route in the tree (i.e., they do not travel in two directions over the same undirected edge), and that each container observed is on its way from one

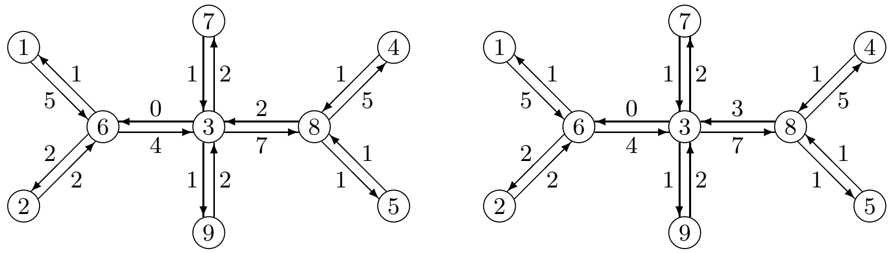

Fig. 2. Two weighted trees that do not have a matching transport

leaf to another (i.e., its origin or destination is not an internal node), then (1) must hold. Each container travelling from a leaf in LLeaves(i, j) to a leaf in RLeaves(i, j) passes edge (i, j) exactly once, and there are no other containers travelling along this (directed) edge.

For example, a matching transport Tr for the tree in Fig. 1 is given by

$$\mathrm{Tr}(1,2) = 2, \mathrm{Tr}(1,4) = 2, \mathrm{Tr}(1,5) = 1, \mathrm{Tr}(2,1) = 1, \mathrm{Tr}(2,9) = 1,$$
$$\mathrm{Tr}(4,7) = 1, \mathrm{Tr}(5,7) = 1, \mathrm{Tr}(7,4) = 1, \mathrm{Tr}(9,4) = 2$$

(and $\mathrm{Tr}(l_1, l_2) = 0$ for pairs of leaves (l_1, l_2) not mentioned).

Now, the problem High Spies can be rephrased as follows: given a weighted tree for which at least one matching transport exists, and given two different leaves l_1 and l_2 of this tree, determine the minimum value and the maximum value for $\mathrm{Tr}(l_1, l_2)$ over all matching transports Tr on the tree.

3 Existence of a Matching Transport

Before we start solving High Spies, we consider the question under which conditions a weighted tree actually permits a matching transport. We have already seen a matching transport for the weighted tree from Fig. 1. If we slightly modify the tree, then there may not exist a matching transport. For example, there do not exist matching transports for the two weighted trees in Fig. 2, in which we have only altered the weights on edges between nodes 3 and 8.

It is intuitively clear that there cannot be a matching transport for the left tree, becauses nodes 3 and 8 do not satisfy the graph analogue of Kirchhoff's current law: the total weight on the edges entering node 3 is unequal to the total weight on the edges leaving that node, and similarly for node 8.

Although the right tree does satisfy Kirchhoff's law, it is also intuitively clear that there cannot be a matching transport for that tree. The weight 7 on edge $(3, 8)$ should be carried along to nodes 4 and 5. However, the total weight on the edges $(8, 4)$ and $(8, 5)$ is only 6. Also, the weight 3 on edge $(8, 3)$ must have arrived at node 8 from nodes 4 and 5. However, the total weight on the edges $(4, 8)$ and $(5, 8)$ is only 2.

In Sect. 4, we prove that these are exactly the types of arguments determining the existence of a matching transport on a weighted tree. For that purpose, we reformulate the (global) definition of a matching transport into local terms.

Definition 4. *Let (T, c) be a weighted tree, and let j be an internal node of T. Let $h_1, \ldots, h_m$ for some $m \geq 2$ be the neighbours of j in T. Then j is a transport node, if there exists an $m \times m$ matrix $A = (a_{i,k})$ satisfying*

$$a_{i,k} \geq 0 \qquad (i, k = 1, \ldots, m) \tag{2}$$

$$a_{i,i} = 0 \qquad (i = 1, \ldots, m) \tag{3}$$

$$\textstyle\sum_{k=1}^{m} a_{i,k} = c(h_i, j) \qquad (i = 1, \ldots, m) \tag{4}$$

$$\textstyle\sum_{i=1}^{m} a_{i,k} = c(j, h_k) \qquad (k = 1, \ldots, m)\ . \tag{5}$$

In this case, the matrix A is called a witness matrix for node j.

Intuitively, a witness matrix determines the transport on a local scale. The entry $a_{i,k}$ can be considered as the number of containers that is shipped from node h_i to node h_k (via node j).

For example, let us consider node $j = 6$ in the tree from Fig. 1. This node, whose neighbours are nodes 1, 2 and 3, is a transport node. If we let $h_1 = 1$, $h_2 = 2$ and $h_3 = 3$, then a witness matrix is

$$A = \begin{array}{ccc} \left(\begin{array}{ccc} 0 & 2 & 3 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{array}\right) & \begin{array}{l} c(1,6) = 5 \\ c(2,6) = 2 \\ c(3,6) = 0 \end{array} \\ \begin{array}{ccc} c(6,1) = 1 & c(6,2) = 2 & c(6,3) = 4 \end{array} & \end{array}$$

Indeed, when we calculate the sums of the individual rows and columns of A, we obtain the weights of the corresponding edges in the tree, as indicated to the right of the matrix and below the matrix.

We now establish that local properties guarantee the existence of a (global) matching transport:

Theorem 5. *Let (T, c) be a weighted tree. There exists a matching transport on (T, c), if and only if each internal node of (T, c) is a transport node.*

Proof. $\Longrightarrow$ Assume that there exists a matching transport Tr on (T, c). Then let j be an arbitrary internal node of T and let $h_1, \ldots, h_m$ for some $m \geq 2$ be the neighbours of j in T. Then the $m \times m$ matrix $A = (a_{i,k})$ defined by

$$a_{i,i} = 0 \qquad (i = 1, \ldots, m)$$

$$a_{i,k} = \sum_{l_1 \in \mathrm{LLeaves}(h_i, j)} \ \sum_{l_2 \in \mathrm{RLeaves}(j, h_k)} \mathrm{Tr}(l_1, l_2) \qquad (i, k = 1, \ldots, m; i \neq k)$$

is a witness for j being a transport node.

$\Longleftarrow$ Assume that each internal node of (T, c) is a transport node. We use induction on the number p of internal nodes to prove that there exists a matching transport Tr on (T, c).

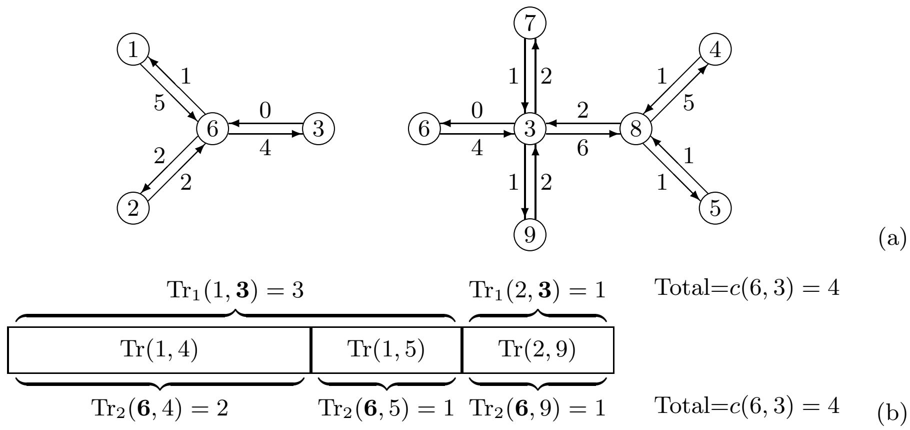

Fig. 3. Constructions used in the proof of Theorem 5. (a) Subtrees T_1 (left) and T_2 (right) resulting when we break up the weighted tree from Fig. 1. (b) Greedy distribution of values $\mathrm{Tr}_1(l_1, 3)$ and $\mathrm{Tr}_2(6, l_2)$ over values $\mathrm{Tr}(l_1, l_2)$.

If $p = 1$, then T is a ‘star graph’. Let j be the only internal node, and let $h_1, \ldots, h_m$ for some $m \geq 2$ be the neighbours of j. These neighbours are exactly all leaves of the tree. By assumption, j is a transport node. The $m \times m$ matrix $A = (a_{i,k})$ which is a witness for this, directly defines a matching transport:

$$\mathrm{Tr}(h_i, h_k) = a_{i,k} \qquad (i, k = 1, \ldots, m; i \neq k) \ .$$

Induction step. Let $p \geq 1$, and suppose that for every weighted tree (T, c) with at most p internal nodes, each of which is a transport node, there exists a matching transport on (T, c) (induction hypothesis). Now consider a weighted tree (T, c) with $p+1$ internal nodes, each of which is a transport node. Let i and j be two arbitrary adjacent, internal nodes.

We use the edge (i, j) to break up (T, c) into two smaller trees with some overlap. Let T_1 be the subtree of T consisting of node j and all nodes in $\mathrm{Left}(i, j)$, together with the edges connecting these nodes. Let T_2 be the subtree of T consisting of node i and all nodes in $\mathrm{Right}(i, j)$, together with the edges connecting these nodes. The weight functions c_1 and c_2 of T_1 and T_2 are equal to c, restricted to T_1 and T_2, respectively. In Fig. 3(a), we have illustrated this construction for edge $(6, 3)$ of the tree from Fig. 1.

Then node j is a leaf in T_1 and T_1 contains at most p internal nodes. Moreover, each internal node in T_1 is also an internal node in T, with the same neighbours and with the same weigths on the edges to and from these neighbours. Consequently, each internal node of T_1 is a transport node, simply because it is one in T. By the induction hypothesis, there exists a matching transport Tr_1 on (T_1, c_1). Analogously, there exists a matching transport Tr_2 on (T_2, c_2).

These two matching transports can be combined into one matching transport Tr on (T, c). For pairs of leaves in T at the same side of edge (i, j), Tr simply inherits the value of Tr_1 or Tr_2. For pairs of leaves in T at different sides of edge (i,j), we observe that (for a transport from left to right)

$$\sum_{l_1 \in \mathrm{LLeaves}(i,j)} \mathrm{Tr}_1(l_1,j) = c(i,j) = \sum_{l_2 \in \mathrm{RLeaves}(i,j)} \mathrm{Tr}_2(i,l_2), \tag{6}$$

as Tr_1 and Tr_2 are matching transports on T_1 and T_2, respectively. We now distribute every value $\mathrm{Tr}_1(l_1,j)$ and every value $\mathrm{Tr}_2(i,l_2)$ over values $\mathrm{Tr}(l_1,l_2)$ (and analogously from right to left). This can be done in a greedy way, as follows.

We take an arbitrary ordering of the values $\mathrm{Tr}_1(l_1,j)$ for $l_1 \in \mathrm{LLeaves}(i,j)$, and partition the total quantity $c(i,j)$ according to these values. We label each resulting fragment with the corresponding leaf l_1. We also take an arbitrary ordering of the values $\mathrm{Tr}_2(i,l_2)$ for $l_2 \in \mathrm{RLeaves}(i,j)$, and partition the same total quantity $c(i,j)$ according to these values, again labelling the fragments with the corresponding leaves. The resulting, double labelling determines Tr.

For example, let us consider matching transports Tr_1 and Tr_2 for the trees T_1 and T_2 in Fig. 3(a), given by

$$\begin{aligned}
&\mathrm{Tr}_1(1,2)=2, \mathrm{Tr}_1(2,1)=1, \mathrm{Tr}_1(1,3)=3, \mathrm{Tr}_1(2,3)=1 \quad \text{and}\\
&\mathrm{Tr}_2(6,4)=2, \mathrm{Tr}_2(6,5)=1, \mathrm{Tr}_2(6,9)=1, \mathrm{Tr}_2(4,7)=1,\\
&\mathrm{Tr}_2(5,7)=1, \mathrm{Tr}_2(7,4)=1, \mathrm{Tr}_2(9,4)=2,
\end{aligned}$$

respectively. The values $\mathrm{Tr}_1(l_1,l_2)$ with $l_1,l_2 \in \{1,2\}$ and $l_1 \neq l_2$ are inherited by Tr, and similarly for $\mathrm{Tr}_2(l_1,l_2)$ with $l_1,l_2 \in \{4,5,7,9\}$ and $l_1 \neq l_2$.

Figure 3(b) illustrates how the values $\mathrm{Tr}_1(1,3)$ and $\mathrm{Tr}_1(2,3)$ (in this order) and the values $\mathrm{Tr}_2(6,4)$, $\mathrm{Tr}_2(6,5)$ and $\mathrm{Tr}_2(6,9)$ (in this order) are distributed over $\mathrm{Tr}(1,4)=2$, $\mathrm{Tr}(1,5)=1$ and $\mathrm{Tr}(2,9)=1$. It should be obvious from the picture and (6) that this algorithm always yields a valid matching transport.

Note that in this example, edge $(3,6)$ has weight 0. so that $\mathrm{Tr}_1(3,l_1)$, $\mathrm{Tr}_2(l_2,6)$ and $\mathrm{Tr}(l_2,l_1)$ must be 0 for all $l_1 \in \mathrm{LLeaves}(6,3)$ and $l_2 \in \mathrm{RLeaves}(6,3)$.

Note also that there usually exist different distributions of $\mathrm{Tr}_1(l_1,j)$ and $\mathrm{Tr}_2(i,l_2)$ over values $\mathrm{Tr}(l_1,l_2)$. For example, if we apply the same greedy algorithm to a different ordering of the values $\mathrm{Tr}_1(l_1,j)$ and/or to a different ordering of the values $\mathrm{Tr}_2(i,l_2)$, then we may obtain a different distribution. □

4 Conditions for Being a Transport Node

By Theorem 5, in order to decide whether or not a matching transport for a weighted tree exists, it suffices to check locally if each internal node is a transport node. By definition, an internal node j is a transport node, if and only if there exists a witness matrix A, which matches the weights $c(i,j)$ and $c(j,i)$ for all neighbours i of j.

Instead of actually constructing such a witness matrix, we now prove that such a witness matrix exists, if and only if the weights $c(i,j)$ and $c(j,i)$ satisfy certain conditions. Then to check if node j is a transport node, we only have to check these conditions.

In this section and in Theorem 11, we consider individual internal nodes j, together with their neighbours and a weight function c on the edges between j and its neighbours. We also use the terms 'transport node' and 'witness matrix', as if they have been defined for this local context. The results that we obtain, however, can be applied in the context of complete weighted trees. We will do that at the end of Sect. 5.

In Definition 4, we denoted the neighbours of an internal node j by $h_1, \ldots, h_m$ for some $m \geq 2$. From now on, for notational convenience, we assume that these neighbours are nodes $1, \ldots, m$, respectively.

Lemma 6. *If node j is a transport node, then*

$$\textstyle\sum_{i=1}^{m} c(i,j) = \sum_{k=1}^{m} c(j,k), \tag{7}$$

$$\textstyle c(i,j) \leq \sum_{k \neq i} c(j,k) \qquad (i = 1, \ldots, m) \quad \textit{and} \tag{8}$$

$$\textstyle c(j,k) \leq \sum_{i \neq k} c(i,j) \qquad (k = 1, \ldots, m) \ . \tag{9}$$

Equation (7) is Kirchhoff's law, meaning that the total weight entering node j equals the total weight leaving node j. Equation (8) expresses the fact that the weight coming in from a neighbour i can go out to the other neighbours of node j. Finally, (9) expresses the fact that the weight going out to a neighbour k can have come from the other neighbours of node j. Exactly these equations were violated by nodes 3 and 8 in the weighted trees in Fig. 2.

Before we prove Lemma 6, we show that there is some redundancy in (7)–(9). For $i_0 = 1, \ldots, m$, let

$$\textstyle \mathrm{Margin}_c(i_0, j) = \sum_{k \neq i_0} c(j,k) - c(i_0, j) \quad \text{and}$$
$$\textstyle \mathrm{Margin}_c(j, i_0) = \sum_{i \neq i_0} c(i,j) - c(j, i_0) \ .$$

Hence, $\mathrm{Margin}_c(i_0, j)$ and $\mathrm{Margin}_c(j, i_0)$ denote the differences between the right hand side and the left side of (8) and (9), respectively. If the weight function c is clear from the context, we will simply write $\mathrm{Margin}(i_0, j)$ and $\mathrm{Margin}(j, i_0)$. We then have:

Lemma 7. *If (7) holds for node j, then for $i_0 = 1, \ldots, m$, Margin(i_0, j) = Margin(j, i_0).*

This result follows directly from the definitions and (7). It implies in particular that if (7) holds, then (8) and (9) are equivalent. Therefore, in the rest of the paper, when we have to prove that (7)–(9) are valid for an internal node j and a weight function c, we will not mention (9).

Proof of Lemma 6. Assume that node j is a transport node. Then by definition, there exists an $m \times m$ matrix $A = (a_{i,k})$ satisfying (2)–(5).

When we add up all entries of A, row by row or column by column, we find

$$\sum_{i=1}^{m} c(i,j) = \sum_{i=1}^{m} \sum_{k=1}^{m} a_{i,k} = \sum_{k=1}^{m} \sum_{i=1}^{m} a_{i,k} = \sum_{k=1}^{m} c(j,k) \ .$$

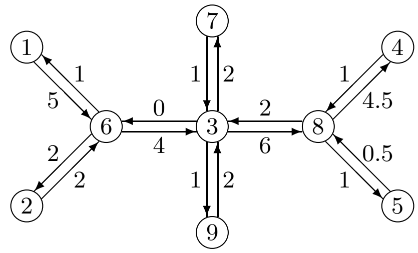

Fig. 4. Tree resulting from the tree in Fig. 1, if we assign a quantity $a = 0.5$ from edge $(5, 8)$ to edge $(8, 4)$

Indeed, (7) holds. Let i_0 be an arbitrary neighbour of j. Then

$$c(i_0, j) = \sum_{k=1}^{m} a_{i_0,k} = \sum_{k \neq i_0} a_{i_0,k} \leq \sum_{k \neq i_0} (\sum_{i=1}^{m} a_{i,k}) = \sum_{k \neq i_0} c(j, k) \ .$$

Hence, also (8) holds. □

At first glance, it is not obvious that Lemma 6 can be reversed. That is, that if (7)–(9) hold for an internal node j, then j is a transport node. In particular, it is imagineable that for each individual neighbour i of j, the weight on edge (i, j) can be carried along to the other neighbours of j, but that this is not possible for all neighbours of j simultaneously. Simply, because the weights on incoming edges (i, j) for different neighbours i are competing for the same outgoing edges (j, k).

For example, let j be node 8 in the tree from Fig. 1. In Fig. 4 we have depicted the tree resulting from assigning a quantity $a = 0.5$ from the incoming edge $(5, 8)$ to the outgoing edge $(8, 4)$. Now, the weight on edge $(3, 8)$ can no longer be passed on to the other neighbours of node 8.

We will see that this problem can be avoided, and that we can indeed reverse Lemma 6. First, however, we state a result on the values of Margin(i, j) for competing edges (i_0, j) and (i_1, j), which follows directly from the definitions and (7).

Lemma 8. *Assume that (7)–(9) hold for node j. Let i_0 and i_1 be two different neighbours of j. Then*

$$\mathit{Margin}(i_0, j) + \mathit{Margin}(i_1, j) = \sum_{i \neq i_0, i_1} c(i, j) + \sum_{k \neq i_0, i_1} c(j, k) \ . \tag{10}$$

We now state the converse of Lemma 6 and provide its proof.

Lemma 9. *If (7)–(9) hold for node j, then j is a transport node.*

Proof. Assume that c satisfies (7)–(9). We prove that we can find an $m \times m$ matrix $A = (a_{i,k})$ satisfying conditions (2)–(5). If this is true, then node j is a transport node.

We use induction on the number p_c of weights on the incoming and outgoing edges of j, which are strictly positive.

If $p_c = 0$, then for $i = 1, \ldots, m$, $c(i, j) = 0$, and for $k = 1, \ldots, m$, $c(j, k) = 0$. It is easily verified that for this case, the $m \times m$ matrix $A = (a_{i,k})$ with all 0-entries satisfies conditions (2)–(5).

Induction step. Let $p \geq 0$, and suppose that for every non-negative weight function c on the incoming and outgoing edges of j, which satisfies (7)–(9) and for which $p_c \leq p$, we can find an $m \times m$ matrix A as specified (induction hypothesis). Now, consider a non-negative weight function c on the same edges and satisfying the same conditions, with $p_c = p + 1$.[2]

Without loss of generality, assume that $c(i_0, j) > 0$ for some neighbour i_0 of j. Let k_0 be an arbitrary neighbour of j which satisfies $k_0 \neq i_0$ and $c(j, k_0) > 0$. By (8), such a neighbour exists.

We now proceed to assign a maximum quantity a of the weight $c(i_0, j)$ of the incoming edge (i_0, j) to the outgoing edge (j, k_0). In order to specify a, we introduce the following quantity:

$$\text{MinMargin} = \min_{i \neq i_0, k_0} \text{Margin}_c(i, j) \ .$$

Let $i_1 \neq i_0, k_0$ be a neighbour of j for which this value is achieved. If $m = 2$, then MinMargin is set to infinity and i_1 is not defined.

Obviously, the value a is bounded by $c(i_0, j)$ and $c(j, k_0)$. It is, however, also bounded by MinMargin. If we let a be larger than MinMargin, then the total weight on outgoing edges, that is available to the incoming edge (i_1, j) would become smaller than $c(i_1, j)$. We therefore set

$$a = \min\big(c(i_0, j), c(j, k_0), \text{MinMargin}\big) \ .$$

We now distinguish two cases:

• If $a = c(i_0, j)$ or $a = c(j, k_0)$, then we define a new weight function c' by subtracting a from $c(i_0, j)$ and $c(j, k_0)$ and leaving all other weights unchanged.

It is easily verified that c' is non-negative and satisfies (7)–(9). Moreover, as either $c'(i_0, j) = 0$, or $c'(j, k_0) = 0$ (or both), $p_{c'} \leq p$. Hence, by the induction hypothesis, there exists an $m \times m$ matrix $A' = (a'_{i,k})$ satisfying conditions (2)–(5) for the weight function c'. It follows immediately that the $m \times m$ matrix $A = (a_{i,k})$ defined by

$$\begin{aligned} a_{i_0,k_0} &= a'_{i_0,k_0} + a \\ a_{i,k} &= a'_{i,k} \qquad\qquad (\ (i,k) \neq (i_0, k_0) \ ) \end{aligned}$$

satisfies the same conditions for the original weight function c.

• If $a = \text{MinMargin}$ and $\text{MinMargin} < \min(c(i_0, j), c(j, k_0))$, then in particular $m \geq 3$ and node i_1 is defined. Now, we define a new weight function c' by

$$c'(i_0, j) = 0$$

[2] Note that, because of (7), it is impossible that $p_c = 1$. This, however, does not harm our argument.

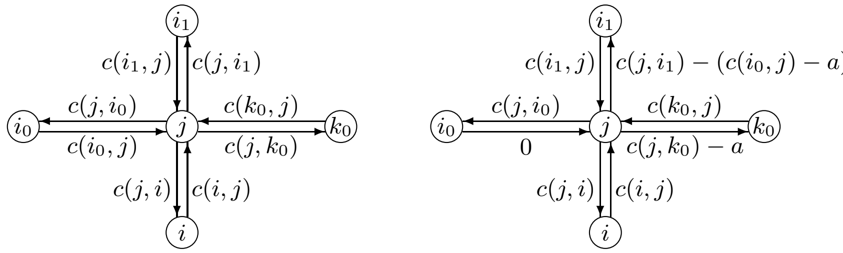

Fig. 5. Old (left) and new (right) weights for the second case in the proof of Lemma 9

$$
\begin{aligned}
c'(j, k_0) &= c(j, k_0) - a \\
c'(j, i_1) &= c(j, i_1) - (c(i_0, j) - a) \\
c'(i, j) &= c(i, j) && (i \neq i_0) \\
c'(j, k) &= c(j, k) && (k \neq k_0, i_1)
\end{aligned}
$$

(see Fig. 5). By definition and by Lemma 7,

$$a = \mathrm{Margin}_c(i_1, j) = \mathrm{Margin}_c(j, i_1) = \sum_{i \neq i_1} c(i, j) - c(j, i_1) \geq c(i_0, j) - c(j, i_1) \ .$$

Hence, $c'(j, i_1) \geq 0$, which implies that c' is non-negative for every edge.

It is easily verified that the weight function c' satisfies (7). It remains to be proved that c' also satisfies (8), i.e., that $\mathrm{Margin}_{c'}(i_2, j) \geq 0$ for each neighbour i_2 of j. For this, we can distinguish four subcases: $i_2 = i_0$, $i_2 = i_1$, $i_2 = k_0$ and $i_2 \neq i_0, i_1, k_0$. The proofs for the first two cases are straightforward. The proofs for the last two cases are more involved, but similar. We give the details for the case that $i_2 \neq i_0, i_1, k_0$:

$$
\begin{aligned}
\mathrm{Margin}_{c'}(i_2, j) &= \sum_{k \neq i_2} c'(j, k) - c'(i_2, j) \\
&= \sum_{k \neq i_2} c(j, k) - a - (c(i_0, j) - a) - c(i_2, j) = \mathrm{Margin}_c(i_2, j) - c(i_0, j) \ .
\end{aligned}
$$

By Lemma 8,

$$
\begin{aligned}
&\mathrm{Margin}_c(i_2, j) + \mathrm{Margin}_c(i_1, j) = \mathrm{Margin}_c(i_2, j) + a \\
&\quad = \sum_{i \neq i_2, i_1} c(i, j) + \sum_{k \neq i_2, i_1} c(j, k) \geq c(i_0, j) + c(j, k_0) > c(i_0, j) + a \ .
\end{aligned}
$$

This implies that $\mathrm{Margin}_{c'}(i_2, j) \geq 0$.

Because by definition, $c'(i_0, j) = 0$, we have $p_{c'} \leq p$. Hence, by the induction hypothesis, there exists an $m \times m$ matrix $A' = (a'_{i,k})$ satisfying conditions (2)–(5) for the weight function c'. It follows immediately that the $m \times m$ matrix

$A = (a_{i,k})$ defined by

$$\begin{aligned} a_{i_0,k_0} &= a'_{i_0,k_0} + a \\ a_{i_0,i_1} &= a'_{i_0,i_1} + (c(i_0,j) - a) \\ a_{i,k} &= a'_{i,k} \qquad\qquad\qquad ((i,k) \neq (i_0,k_0), (i_0,i_1)) \end{aligned}$$

satisfies the same conditions for the original weight function c. □

When we combine Lemma 6 and Lemma 9, we obtain

Theorem 10. *Node j is a transport node, if and only if (7)–(9) hold for j.*

5 Minimum and Maximum Transport

A transport node in a weighted tree may have many different witness matrices. We examine the minimum and maximum values for each of the entries in these witness matrices.

Theorem 11. *Assume that node j is a transport node and that $A = (a_{i,k})$ is a witness matrix for this. Let i_0 and k_0 be two arbitrary, different neighbours of j. Then*

$$a_{i_0,k_0} \geq \max\Big(0,\ c(i_0,j) - \sum_{k \neq i_0,k_0} c(j,k),\ c(j,k_0) - \sum_{i \neq i_0,k_0} c(i,j) \Big) \ \textit{and} \tag{11}$$

$$a_{i_0,k_0} \leq \min\Big(c(i_0,j),\ c(j,k_0),\ \min_{i \neq i_0,k_0} \mathit{Margin}_c(i,j) \Big), \tag{12}$$

and each value satisfying these equations can be achieved.

It would be a nice exercise to prove that the right-hand side of (11) is at most as large as the right-hand side of (12), without using these equations themselves.

Note that by (4) and Lemma 7, (11) is equivalent to the following inequality:

$$\sum_{k \neq k_0} a_{i_0,k} \leq \min\Big(c(i_0,j),\ \sum_{k \neq i_0,k_0} c(j,k),\ \mathrm{Margin}_c(k_0,j) \Big) \ . \tag{13}$$

Proof. We first prove (11). By definition, $a_{i_0,k_0} \geq 0$. By successively applying (3), (2) and (5), we find

$$\sum_{k \neq k_0} a_{i_0,k} = \sum_{k \neq i_0,k_0} a_{i_0,k} \leq \sum_{k \neq i_0,k_0} \sum_{i=1}^{m} a_{i,k} = \sum_{k \neq i_0,k_0} c(j,k) \ .$$

Hence, by (4), $a_{i_0,k_0} \geq c(i_0,j) - \sum_{k \neq i_0,k_0} c(j,k)$. Analogously, we find $a_{i_0,k_0} \geq c(j,k_0) - \sum_{i \neq i_0,k_0} c(i,j)$.

In the proof of Lemma 9, we have seen that also (12) holds.

We now prove that each value $a_{i_0,k_0} = a$ satisfying (11) and (12) can be achieved. If $m = 2$, then $\sum_{k \neq i_0,k_0} c(j,k) = \sum_{i \neq i_0,k_0} c(i,j) = 0$. Hence, by (11) and (12), the only possible value for a_{i_0,k_0} is $a_{i_0,k_0} = c(i_0,j) = c(j,k_0)$. The existence of a witness matrix A implies that this value can indeed be achieved.

Now assume that $m \geq 3$ and let $a_{i_0,k_0} = a$ be an arbitrary value satisfying (11) and (12).

By Lemma 6, we know that the weight function c satisfies (7)–(9). Let the weight function c' be defined by

$$\begin{aligned} c'(i_0,j) &= c(i_0,j) - a \\ c'(j,k_0) &= c(j,k_0) - a \\ c'(i,j) &= c(i,j) && (i \neq i_0) \\ c'(j,k) &= c(j,k) && (k \neq k_0) \ . \end{aligned}$$

Because $a \leq c(i_0,j)$ and $a \leq c(j,k_0)$, c' is non-negative. Further, c' satisfies (7), because c does. Finally, because $a \leq \min_{i \neq i_0,k_0} \mathrm{Margin}_c(i,j)$, $\mathrm{Margin}_{c'}(i,j) \geq 0$ for $i = 1,\ldots,m$. Hence, c' also satisfies (8). By Theorem 10, j is still (in the context of the new weight function c') a transport node.

Now, let $k_1,\ldots,k_{m-2}$ be the neighbours of j different from i_0 and k_0. We must assign the remaining weight $c'(i_0,j) = c(i_0,j) - a$ on the incoming edge (i_0,j) to the outgoing edges $(j,k_1),\ldots(j,k_{m-2})$. That is, we must find a witness matrix $A' = (a'_{i,k})$ for j and c' for which $a'_{i_0,k_0} = 0$.

We can prove that such a matrix exists by induction on the number of neighbours $k \in \{k_1,\ldots,k_{m-2}\}$ for which $\mathrm{Margin}_{c'}(k,j) < c'(i_0,j)$. The intuition behind this is, that if for some neighbour k_l with $1 \leq l \leq m-2$, $\mathrm{Margin}_{c'}(k_l,j) \geq c'(i_0,j)$, then after *any* partitioning of $c'(i_0,j)$ over $a'_{i_0,k_1},\ldots,a'_{i_0,k_{m-2}}$, we can still pass the weight $c'(k_l,j)$ to the edges (j,k) with $k \neq k_l$. If, on the other hand, $\mathrm{Margin}_{c'}(k_l,j) < c'(i_0,j)$, then we must make sure that a'_{i_0,k_l} gets at least a share $c'(i_0,j) - \mathrm{Margin}_{c'}(k_l,j)$ of the weight $c'(i_0,j)$.

As the details of this proof are rather technical, we do not carry out this proof here. We just make two observations. First, by (11),

$$c'(i_0,j) = c(i_0,j) - a \leq \sum_{k \neq i_0,k_0} c(j,k) = \sum_{k \neq i_0,k_0} c'(j,k) \ .$$

Hence, the total weight available on the edges $(j,k_1),\ldots,(j,k_{m-2})$ is enough to receive the remaining weight on edge (i_0,j). Second, by (11) and Lemma 7,

$$\begin{aligned} c'(i_0,j) = c(i_0,j) - a &\leq c(i_0,j) - c(j,k_0) + \sum_{i \neq i_0,k_0} c(i,j) \\ &= \mathrm{Margin}_c(j,k_0) = \mathrm{Margin}_c(k_0,j) = \mathrm{Margin}_{c'}(k_0,j) \ . \end{aligned}$$

Hence, after assigning $c'(i_0,j)$ to the edges $(j,k_1),\ldots,(j,k_{m-2})$, we can still pass the weight $c'(k_0,j)$ of edge (k_0,j) to neighbours of j different from k_0.

Assuming that we can find the matrix A', the matrix $A^* = (a^*_{i,k})$ defined by

$$\begin{aligned} a^*_{i_0,k_0} &= a \\ a^*_{i,k} &= a'_{i,k} \qquad (\ (i,k) \neq (i_0,k_0)\ ) \end{aligned}$$

is a witness matrix for j and c, with the desired value a for $a^*_{i_0,k_0}$. □

Algorithm for the Global Problem

We return to the context of a complete weighted tree (T, c). We can use Theorem 5 and Theorem 10 to decide whether or not there exists a matching transport Tr on (T, c). Assume that this is the case. For two different leaves l_1 (the source) and l_2 (the target), let $\mathrm{MinTr}(l_1, l_2)$ and $\mathrm{MaxTr}(l_1, l_2)$ denote the minimum and maximum possible values for $\mathrm{Tr}(l_1, l_2)$ respectively. We use Theorem 11 to determine these values. The algorithm for this is simple.

The transport from l_1 to l_2 follows the (unique) path in the tree from l_1 to l_2. Let $j_1, \ldots, j_p$ for some $p \geq 0$ be the internal nodes on this path, in the order of occurrence on the path. Let us define $j_0 = l_1$ and $j_{p+1} = l_2$.

If $p = 0$ (which is only possible if l_1 and l_2 are the only nodes in the tree), then obviously, $\mathrm{MinTr}(l_1, l_2) = \mathrm{MaxTr}(l_1, l_2) = c(l_1, l_2)$. It is impossible for containers from l_1 to 'escape' to other destinations, as there are no other destinations.

Now assume that $p \geq 1$. Let us denote the lower bound and upper bound for a_{i_0,k_0} from Theorem 11 by $\mathrm{LB}(i_0, j, k_0)$ and $\mathrm{UB}(i_0, j, k_0)$, respectively.

As announced in the introduction, to obtain $\mathrm{MinTr}(l_1, l_2)$, we proceed in a greedy way. In each internal node j_k on the path from l_1 to l_2, we direct as much weight as possible from the preceding node j_{k-1} to neighbours of j_k other than j_{k+1}. As we have to pass at least $\mathrm{LB}(j_{k-1}, j_k, j_{k+1})$ of the weight from j_{k-1} to j_{k+1}, we can direct at most $c(j_{k-1}, j_k) - \mathrm{LB}(j_{k-1}, j_k, j_{k+1})$ to the other neighbours (see also (13)). Then $\mathrm{MinTr}(l_1, l_2)$ equals the weight remaining from $c(l_1, j_1)$ after passing by all internal nodes:

$$\mathrm{MinTr}(l_1, l_2) = \max\Big(\, 0,\ c(l_1, j_1) - \sum_{k=1}^{p} \big(c(j_{k-1}, j_k) - \mathrm{LB}(j_{k-1}, j_k, j_{k+1})\big)\,\Big)\,. \quad (14)$$

To obtain $\mathrm{MaxTr}(l_1, l_2)$, for each internal node j_k on the path from l_1 to l_2, we pass as much weight as possible from j_{k-1} via j_k to j_{k+1}. The minimum value $\mathrm{UB}(j_{k-1}, j_k, j_{k+1})$ we encounter on the path determines the maximum weight that can be transported from l_1 to l_2:

$$\mathrm{MaxTr}(l_1, l_2) = \min_{k=1}^{p} \mathrm{UB}(j_{k-1}, j_k, j_{k+1})\ . \quad (15)$$

The complete algorithm consists of first determining the path from l_1 to l_2, and then (if $p \geq 1$) calculating (14) and (15). It is not hard to see that the time complexity of this algorithm is linear in the size of the input.

Let us consider the weighted tree from Fig. 1, and let $l_1 = 1$ and $l_2 = 4$. Then the path from l_1 to l_2 contains $p = 3$ internal nodes 6, 3, 8, and our algorithm yields the following sequence of values:

k	0	1	2	3	4
j_k	1	6	3	8	4
$c(j_{k-1}, j_k) - \mathrm{LB}(j_{k-1}, j_k, j_{k+1})$		2	1	1	
$\mathrm{UB}(j_{k-1}, j_k, j_{k+1})$		3	4	5	

As a result, $\mathrm{MinTr}(1, 4) = 5 - (2+1+1) = 1$ and $\mathrm{MaxTr}(1, 4) = \min(3, 4, 5) = 3$.

6 Concluding Remarks

A large fraction of this paper dealt with transport nodes and corresponding witness matrices $A = (a_{i,k})$. In the resulting algorithm, these concepts do not play any role. However, we needed them to prove that the algorithm is correct, i.e., to prove that the minimum (or maximum) value computed for the transport from one leaf (the source) to another leaf (the target) in a weighted tree, can be extended to a complete matching transport on the tree, and that this value is indeed minimal (maximal, respectively).

In the introduction, we explained why standard max-flow algorithms such as the Ford-Fulkerson algorithm could not be applied directly to High Spies. For the integer-valued case, however, there does exist an approach to the problem, based on such algorithms. In this approach, the lower bound and upper bound for the (local) transport at an internal node j are found by fixing a candidate value $a_{i_0,k_0} = a$ (for proper choices of a), and trying to extend this to a complete $m \times m$ witness matrix $A = (a_{i,k})$. With two copies of every neighbour i of j (corresponding to the edges (i, j) and (j, i), respectively), two additional nodes (source and target) and proper connections between the nodes, A can be derived from a maximum flow with value $\sum_i^m c_{i,j} - a$. However, as the number of connections is quadratic in m and the complexity of max-flow algorithms is at least linear in this number, this 'max-flow approach' is far more time-consuming than applying Theorem 11.

The 'max-flow approach' can be extended in a natural way to an approach for finding the maximum (global) transport between two leaves, which does not rely on (local) transports at internal nodes. Again, however, this would be far less efficient than our algorithm. Moreover, this extension does not work for the minimum (global) transport.

Both the original definition of a matching transport (Definition 3) and the equivalent formulation in terms of transport nodes from Theorem 5 are well suited to generate instances of High Spies. Due to space limitations, we could not elaborate on that here, but we plan to do this in a forthcoming report. In that report, we will also give more detailed proofs for some of the results in this paper.

We want to emphasize, that we did not expect the teams that participated in the Benelux Algorithm Programming Contest 2006 to prove that their solutions for High Spies were correct. A correct implementation of the algorithm described in Sect. 5 was sufficient.

References

Ford Jr., L.R., Fulkerson, D.R.: A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. Canadian Journal of Mathematics 9, 210–218 (1957)

Shasha, D.E.: Puzzling adventures: High spies. Scientific American 289(1), 80 (July 2003)